Glisser-Déposer avec PyQT6

Peter Philippe

23/06/2025

Introduction

Cette succinte note propose une ébauche d'interface pour un logiciel de commande d'une pizzeria. Ce minuscule programme pourra éventuellement servir de base en vue de l'écriture d'un programme nettement plus élaboré, avec notamment l'ajout de boutons + et - permettant respectivement de dupliquer et supprimer un article, d'effectuer un export en PDF avec le nom du client et ses coordonnées pour la livraison (enregistrement des comptes dans une base de données avec `QtSql`), d'ajouter la possibilité d'éditer les prix depuis une fenêtre accessible après avoir validé un mot de passe, de proposer le choix du diamètre et d'autres options comme un supplément pour une pâte épaisse pour chaque pizza, mais encore d'augmenter la liste des articles disponibles en ajoutant notamment des fruits, des soupes de légumes, des pistaches, des paninis, des barquettes de frites, des glaces, etc.).

Bon, arrêtons d'emblée de nous lécher les babines [1] pour saisir quelques lignes de code (à partir de la page 4) afin de reproduire l'interface affichée à la page suivante :

1. Ne pas jamais oublier que si l'on souhaite faire de vieux os, les graisses saturées, les sucres rapides (hormis ceux des fruits) et le sel font partie de nos pires ennemis sur terre.

Le montant est de 23,50 € car l'article *Tiramisu* au prix de 6,50 € n'a pas encore été déposé dans la liste des articles à commander. La capture ci-dessous montre le total de 30 € après avoir déposé l'article *Tiramisu* :

1 Quelques explications sur le code source

On rappelle qu'utiliser PyQT6 ou PySide6 revient au même à 99,9999...%. Une classe fondamentale de QT est `QObject`, tous les objets de QT reposent sur elle (et ça fait du monde !). Voici les imports de notre programme où j'ai choisi délibérément, à tord ou à raison, de ne pas utiliser `QDrag` de `PyQt6.QtGui`.

La classe `PyQt6.QtCore` permet la gestion pour des entrées-sorties comme celles des fichiers et de la caméra, mais aussi le contrôle du timer, des nombres aléatoires, des strings et des expressions régulières, le multithreading, les signaux, etc. :

Qt : propose notamment la méthode `ItemDataRole` pour gérer les types personnalisés.

QIODevice : fournit une interface en lecture-écriture pour les données.

QDataStream : permet la sérialisation binaire dans un format indépendant de l'OS.

Et la classe `PyQt6.QtWidgets` est une classe très importante, puisqu'elle est LA classe mère de tous les composants d'une interface graphique, notamment ceux avec lesquels on interagit avec la souris :

QWidget : classe de base des éléments d'une interface.

QApplication : indispensable pour toutes les applications à base de QWidgets.

QGridLayout : grille uniforme pour placer les QWidgets (un exemple classique est celui d'une calculatrice).

QVBoxLayout : boîte englobante horizontale pour accueillir des QWidgets.

QHBoxLayout : boîte englobante verticale accueillir pour des QWidgets.

QLabel : libellé de texte.

QListWidget : liste d'éléments.

QListWidgetItem : éléments de la liste ci-dessus.

QPushButton : bouton poussoir.

QFrame : souvent utilisé comme un cadre pour entourer un élément.

Concernant les classes, il n'y en a que 3 :

ListeArticles : création d'une liste contenant les articles proposés.

ListeCommandeArticles : création d'une liste contenant les articles à commander.

PizzeriaBase : fenêtre de base de notre application.

Les fonctions `dragEnterEvent`, `dragMoveEvent`, `dropEvent` sont internes à QT6 et permettent respectivement de gérer le glisser, le déplacement et le déposer depuis les trois listes vers la commande. Car ici, seule la liste de la commande doit recevoir des articles via le glisser-déposer, les trois autres listes *Pizzas*, *Boissons* et *Desserts* doivent ignorer les évènement avec `dragEnterEvent`, `dropEvent`, `setAcceptDrops(False)` et `setDragDropMode(QListWidget.DragDropMode.DragOnly)`.

Ici, bien que cela ne soit pas conseillé, toutes les variables sont dans la langue de Molière, cela facilite quelque peu la compréhension, il est par contre fortement conseillé de ne jamais procéder de la sorte en tant que professionnel(le).

2 Code source

Ce code est extrait d'un petit programme (avec la plupart des ajouts cités en introduction) que j'avais commencé à écrire avec PyQT4 [2], en m'imaginant justement travailler dans une pizzeria :

```
import sys
import uuid
from PyQt6.QtCore import Qt, QIODevice, QDataStream
from PyQt6.QtWidgets import (
    QWidget, QApplication, QGridLayout, QVBoxLayout, QHBoxLayout,
    QLabel, QListWidget, QListWidgetItem, QPushButton, QFrame
)

PRIX_PAR_DEFAUT = {
    "PIZZA": {
        "Reine": 12.0, "Napolitaine": 13.0, "Marguerite": 14.0,
        "Cannibale": 19.0, "Norvégienne": 22.0, "Truffe": 25.0
    },
    "BOISSONS": {"Eau plate": 1.5, "Eau gazeuse": 2.0, "Jus de fruits maison": 4.5},
    "DESSERTS": {"Tarte aux pommes": 4.5, "Muffin au chocolat": 4.0, "Tropézienne": 5.5,
                 "Tiramisu": 6.5}
}

class ListeArticles(QListWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setAcceptDrops(False)
        self.setDragDropMode(QListWidget.DragDropMode.DragOnly)
```

2. Il y a donc de cela quelques années..., j'ai extrait ces quelques lignes et les ai adapté pour PyQT6.

```
    def dragEnterEvent(self, event):
        event.ignore()

    def dropEvent(self, event):
        event.ignore()


class ListeCommandeArticles(QListWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setAcceptDrops(True)
        self.setDragDropMode(QListWidget.DragDropMode.DropOnly)
        self.itemDropped = None
        self.parent_app = parent

        self.setStyleSheet("""
            QListWidget {
                background-color: #FFFFFF;
                border: 2px solid #FCC;
                border-radius: 10px;
                padding: 5px;
            }
        """)

    def dragEnterEvent(self, event):
        if event.mimeData().hasFormat("application/x-qabstractitemmodeldatalist"):
            event.accept()
        else:
            event.ignore()

    def dragMoveEvent(self, event):
        if event.mimeData().hasFormat("application/x-qabstractitemmodeldatalist"):
            event.setDropAction(Qt.DropAction.CopyAction)
            event.accept()
        else:
            event.ignore()

    def dropEvent(self, event):
        # le format "QMimeData" est utilisé pour copier des données vers le presse-papier
        # ou bien pour les déplacer via le glisser-déposer.
        mime_data = event.mimeData()
        # https://doc.qt.io/qtforpython-6/PySide6/QtCore/QAbstractItemModel.html
        if mime_data.hasFormat("application/x-qabstractitemmodeldatalist"):
            encoded_data = mime_data.data("application/x-qabstractitemmodeldatalist")
            # https://doc.qt.io/qtforpython-6/PySide6/QtCore/QDataStream.html
            data_stream = QDataStream(encoded_data, QIODevice.OpenModeFlag.ReadOnly)
            data_stream.setVersion(QDataStream.Version.Qt_6_0)

            while not data_stream.atEnd():
                _ = data_stream.readInt()
```

```
                _ = data_stream.readInt()
                map_items = data_stream.readInt()

                for _ in range(map_items):
                    cle = data_stream.readInt()
                    valeur = data_stream.readQVariant()

                    if Qt.ItemDataRole(cle) == Qt.ItemDataRole.UserRole:
                        categorie, libelle, prix = valeur
                        prix_article = PRIX_PAR_DEFAUT.get(categorie, {}).get(libelle, prix)
                        texte_formate = f"{libelle} - {prix_article:.2f} €"
                        article_depose = QListWidgetItem(texte_formate)
                        article_a_commander = (categorie, libelle, prix_article)
                        if categorie == "PIZZA":
                            pizza_identifiant = str(uuid.uuid4()) # ID aléatoire
                            article_a_commander = (categorie, libelle, prix_article, pizza_identifiant)
                            self.parent_app.dernier_element_ajoute = article_depose
                        article_depose.setData(Qt.ItemDataRole.UserRole, article_a_commander)
                        self.addItem(article_depose)
                        if self.itemDropped:
                            self.itemDropped()
            event.acceptProposedAction()
        else:
            event.ignore()

class PizzeriaBase(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Commande Pizzeria : démo glisser-déposer")
        self.prix = PRIX_PAR_DEFAUT
        self.initialiser_interface()
        self.setStyleSheet(self.style_global())

    def style_global(self):
        return """
            QWidget {
                background-color: #FF6347;
                color: #333;
                font-family: Arial, sans-serif;
            }
            QLabel {
                color: #FFFFFF;
            }
            QLabel#montant_libelle {
                font-size: 20px;
                font-weight: bold;
                color: #FFFFFF;
            }
            QListWidget {
                background-color: #FFFFFF;
                border: 1px solid #ddd;
```

```
            border-radius: 5px;
            padding: 5px;
            color: #333;
        }
        QListWidget:hover {
            background-color: #FF8569;
            border: 1px solid #fff;
            border-radius: 5px;
        }
        QListWidget::item {
            padding: 3px;
        }
        QListWidget::item:selected {
            background-color: #FF6347;
            color: white;
        }
        QPushButton {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            font-size: 18px;
            margin: 5px;
        }
        QPushButton:hover {
            background-color: #45a049;
            border: 1px solid #fff;
            border-radius: 5px;
        }
        QFrame {
            background-color: transparent;
        }
        QFrame[frameShape="4"] {
            background-color: #FFDAB9;
            height: 2px;
        }
    """

def creer_liste_par_categorie(self, titre, articles):
    layout = QVBoxLayout()
    layout.addWidget(QLabel(f"<b>{titre}</b>"))
    liste_widgets_articles = ListeArticles()
    liste_widgets_articles.setDragEnabled(True)

    for nom in articles:
        prix = PRIX_PAR_DEFAUT[titre][nom]
        article = QListWidgetItem(f"{nom} - {prix:.2f} € ")
        article.setData(Qt.ItemDataRole.UserRole, (titre, nom, prix))
        article.setFlags(article.flags() | Qt.ItemFlag.ItemIsDragEnabled)
        liste_widgets_articles.addItem(article)
```

```
        layout.addWidget(liste_widgets_articles)
        return layout, liste_widgets_articles

    def initialiser_interface(self):
        layout_base = QHBoxLayout()
        self.listes = {}
        categorie_prix = list(PRIX_PAR_DEFAUT.keys())

        layout_categorie = QVBoxLayout()
        layout_grille = QGridLayout()
        ligne = 0
        colonne = 0
        for categorie in categorie_prix:
            layout, liste_widgets = self.creer_liste_par_categorie(categorie, self.prix[categorie])
            layout_grille.addLayout(layout, ligne, colonne)
            self.listes[categorie] = liste_widgets
            colonne += 1
            if colonne > 2:
                colonne = 0
                ligne += 1
        layout_categorie.addLayout(layout_grille)
        layout_base.addLayout(layout_categorie)

        layout_commande = QVBoxLayout()
        layout_commande.addWidget(QLabel("<b>Commande</b>"))

        self.liste_commande = ListeCommandeArticles(self)
        self.liste_commande.itemDropped = self.mettre_a_jour_montant
        self.liste_commande.itemClicked.connect(self.dernier_element_clique)
        layout_commande.addWidget(self.liste_commande)

        self.montant_libelle = QLabel("Montant : 0.00 €")
        self.montant_libelle.setObjectName("montant_libelle")
        self.montant_libelle.setAlignment(Qt.AlignmentFlag.AlignRight | Qt.AlignmentFlag.AlignVCenter)
        layout_commande.addWidget(self.montant_libelle)

        separateur = QFrame()
        separateur.setFrameShape(QFrame.Shape.HLine)
        separateur.setFrameShadow(QFrame.Shadow.Sunken)
        layout_commande.addWidget(separateur)

        bouton_annuler = QPushButton("Annuler la commande")
        bouton_annuler.clicked.connect(self.annuler_commande)
        layout_commande.addWidget(bouton_annuler)
        layout_commande.addStretch(1)
        layout_base.addLayout(layout_commande)
        self.setLayout(layout_base)
        self.setFixedSize(850, 400)

    def dernier_element_clique(self, item):
        item_data = item.data(Qt.ItemDataRole.UserRole)
```

```
        if item_data and item_data[0] == "PIZZA":
            self.dernier_element_ajoute = item

    def annuler_commande(self):
        self.liste_commande.clear()
        self.dernier_element_ajoute = None
        self.mettre_a_jour_montant()

    def mettre_a_jour_montant(self):
        total = 0.0
        for i in range(self.liste_commande.count()):
            article = self.liste_commande.item(i)
            article_data = article.data(Qt.ItemDataRole.UserRole)
            if article_data and isinstance(article_data, tuple) and len(article_data) >= 2:
                category = article_data[0]
                name = article_data[1]
                prix_article_ajoute = PRIX_PAR_DEFAUT.get(category, {}).get(name, 0.0)
                total += prix_article_ajoute

        self.montant_libelle.setText(f"Montant : {total:.2f} € ")


if __name__ == "__main__":
    app = QApplication(sys.argv)
    fenetre = PizzeriaBase()
    fenetre.show()
    sys.exit(app.exec())
```